Formal Validation of Probabilistic Collision Risk Estimation for Autonomous Driving

Philippe Ledent, Anshul Paigwar, Alessandro Renzaglia, Radu Mateescu, Christian Laugier

Formal Validation of Probabilistic Collision Risk Estimation for Autonomous Driving

Philippe Ledent, Anshul Paigwar, Alessandro Renzaglia, Radu Mateescu and Christian Laugier

Abstract— **Autonomous driving technology is rapidly advancing towards level 5 autonomy along with claims of increasing safety on roads. However, a proper validation of such safety-critical, complex systems and of their reliability still needs to be addressed adequately. To this end, standard exhaustive methods are inappropriate to validate the probabilistic algorithms widely used in this field and new solutions need to be adopted. In this work, we present a new approach where formal verification is employed to validate systems with probabilistic predictions. In particular, we focus on the risk assessment generated by a probabilistic perception system, the Conditional Monte Carlo Dense Occupancy Tracker (CMCDOT). This framework provides an environment representation through Bayesian probabilistic occupancy grids and estimates Time-to-Collision probabilities for every static and dynamic part of the grid in near future. Focusing on the validation of the probabilistic collision risk estimation, we adopt the CARLA simulator to generate a large number of realistic intersection crossing scenarios with two vehicles. The formal verification is then performed using the XTL model checker of the CADP toolbox, based on the definition of appropriate Key Performance Indicators (KPIs). Finally, a quantitative analysis that goes beyond classical temporal logic verification is provided.**

I. Introduction

The use of formal methods for verification and certification of algorithms and mathematical logic of software is a long standing research [1]. While our software are continuously evolving, becoming more capable and complex, the methods to verify software need to evolve simultaneously. Physical and software components are getting deeply intertwined, each operating on different spatial and temporal scales. Verification of such automated cyber-physical systems (ACPS) is still an open research challenge.

Autonomous vehicles represent a critical ACPS that require meticulous attention in validation. Moreover, there are increasing demands on regulating and validating intelligent vehicle systems to build public trust in their use. Formal methods have already had several successful deployments in fields like aerospace and railways [2], but autonomous driving remains a challenging field because of the complexity of its key components, such as: perception of the environment, scene interpretation, decision making and undertaking actions. Considering the fact that intelligent vehicles are supposed to be driven on existing roads, along with other vehicles with human drivers, a higher level of uncertainties need to be taken into account [3]. In these scenarios, the use of probabilistic approaches for perception and prediction become essential to accurately confront the uncertainties in the environment [4]. The stochasticity of these algorithms, involving multiple states and complex transitions between them, makes the validation through standard approaches often unviable.

Most of the existing literature revolves around the formal verification of decision algorithms [5], [6]. Researchers are exploring formal verification techniques for Probabilistic Model Checking [7]. Compared to conventional systems (e.g., discrete-time Markov models), where the formal model accurately reflects the actual behaviour of the real-world system [8], verification through formal model is fairly accurate but it becomes more challenging for autonomous systems in dynamic environments. In the context of intelligent vehicle systems, where states (e.g., estimation of collisions) change and evolve at every time step, exhaustively checking such properties is usually not affordable in many scenarios because of time, complexity and costs constraints.

The main contribution of this paper is a methodology to validate perception systems based on a combination of simulation, formal verification, and statistical analysis. The simulation framework must comprise a realistic description of the dynamic physical environment (e.g., urban landscape and vehicles), the perception system under study, and a set of relevant scenarios (e.g., leading to collisions). We consider here the validation of the Conditional Monte Carlo Dense Occupancy Tracker (CMCDOT) [9], in particular its collision risk estimation functionality. Executing the simulation scenarios yields traces of events containing the suitable data for validation (timestamp, estimated probabilities of collision, position and velocities of the vehicles, etc.). Then, on each trace obtained, a number of typical temporal properties (invariants, safety, liveness) are verified using the XTL [10] model checker of the CADP toolbox [11], producing quantitative verdicts (sets of events violating the properties, with their diagnostic information). Finally, a statistical analysis of the verdicts is carried out, by defining an appropriate Key Performance Indicator (KPI) for each property, using it to compute a grade for each trace, and aggregating the results in analysis reports using R-studio. The whole methodology has been automated and can be instantiated in other contexts as well.

The rest of the paper is organized as follows. Section II briefly introduces the main properties of CMCDOT, with a particular focus on the collision risk estimation. Section III discusses the principles of formal validation and how it is employed on the considering problem. In Section IV the results obtained with the analysis of intersection crossing scenarios are finally presented and discussed.

Univ. Grenoble Alpes, Inria, 38000, Grenoble, France; e-mail: firstname.lastname@inria.fr

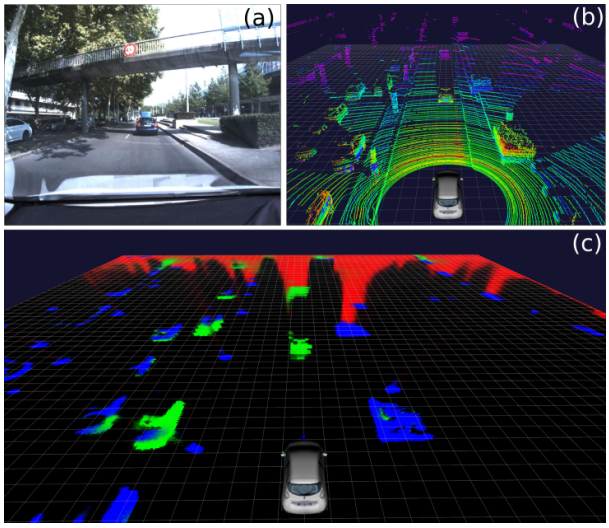

Fig. 1: (Top) Real environment around the car and as observed using a 64 layered lidar. (Bottom) Representation of the environment as dense probabilistic occupancy grid output from CMCDOT framework. Colors represents different state of each cell in the grid: occupied static (blue), occupied dynamic (green), empty (black), unknown area (red).

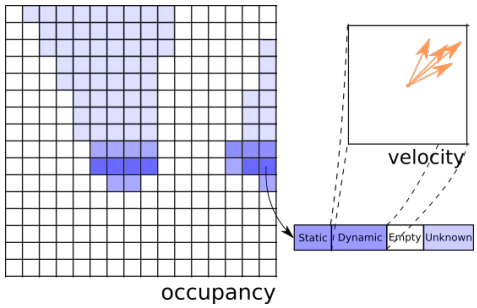

Fig. 2: The CMCDOT represents the environment as a grid, to whose cells are associated static, dynamic, empty and unknown coefficients. Weighted particles, which sample the velocity space, are then associated to the dynamic part.

II. Collision Risk Estimation

The CMCDOT framework [9] is a perception system which provides a dense and generic representation of the environment [12], [13] as a probabilistic occupancy grid (see Fig. 1), based on Bayesian fusion, filtering of sensor data and Bayesian inference [14]. By exploiting specific sensor models, the gathered data are converted in probabilistic estimations which represent the environments as static and dynamic occupied regions, free spaces and unknown areas (Fig. 2). This differentiation enables the use of state-specific models, such as classic occupancy grids for static components and sets of moving particles for dynamic occupancy, as well as confidence estimation and management of areas with no information.

The resulting approach is particularly suitable in situations where incomplete or even contradictory data come from different sensors (camera, lidars, radars, etc.). The estimation

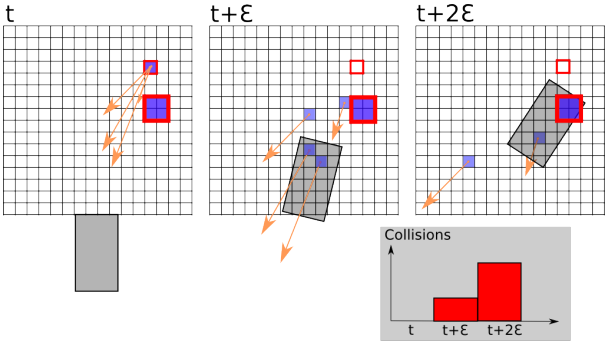

Fig. 3: Collision risk estimation of a specific cell. Both the future cell and the ego-vehicle positions are predicted according to their estimated velocity. The risk of every cell is used to integrate over time the total collision risk.

for each cell over time can be obtained from various sensors data, whose specific uncertainty (noise, measurement errors) is taken into consideration. Filtered cell estimations are thus much more robust, leading to a more reliable global occupancy of the environment, reducing false detection.

Additionally, an important feature of the CMCDOT, and the main subject of study of this paper, is the estimation of the collision risk for each cell of the grid. Most of the existing risk estimation methods consist in detecting and tracking dynamic objects in the scene [15], [16]. The simplest solution is then to estimate a Time to Collision (TTC) by projecting the ego-vehicle and the trajectories of other moving object in the future [17]. Detection of moving objects requires to pre-define shape of objects and involves fitting of object models in pointcloud to find the location of objects in environment. As the number and types of objects increase, the detection and tracking becomes difficult and costly problem to solve.

The grid-based approach used in the CMCDOT framework, instead of detecting objects, directly computes estimations of the position of every static and dynamic cell of the grid by linearly projecting them in near future based upon their estimated velocity as well as the trajectory of the ego-vehicle. These estimations are iteratively computed over short-time periods, until a potential collision is detected. In this case a TTC is associated to the cell from which the colliding element came from (Fig. 3). The probabilistic estimation for different TTCs (e.g, 1, 2 and 3 seconds) are then associated to every cell to obtain a collision risk profile. This strategy, originally presented in [18], presents the advantage of being model free, avoids solving the complex problem of multi-object detection and tracking, while integrating the totality of the available information and providing a probabilistic estimation of the risk associated to each part of the scene. The result of this estimation can then be used as the starting point for any control and decision-making system.

A. Simulation for perception

In this work, the simulation relies on the use of two frameworks: CARLA, an open urban driving simulator [19],

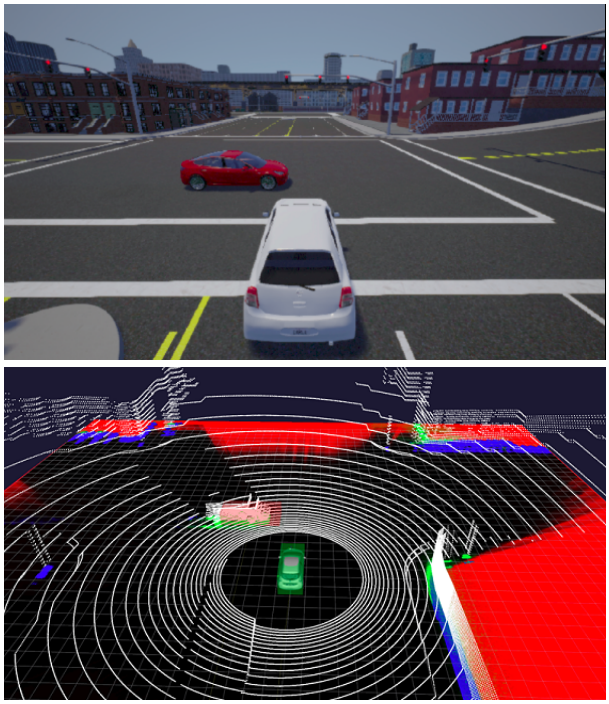

Fig. 4: Simulated scenario in Carla (top) and output of CMCDOT (bottom).

and Robot Operating System (ROS). CARLA simulation environment consists of complex urban layouts, buildings and vehicles rendered in high quality, allowing for a realistic representation of real-world scenarios. The ego-vehicle and its sensors, as well as other moving vehicles, as depicted in Figure 4, can be configured in the simulation to match with the actual system. The provided CARLA-ROS bridge enables data acquisition from the simulation in native ROS message formats, where the data can be recorded, stored, and processed by the same code running on the actual vehicle.

In order to establish the ground truth, a grid indicating the position of all simulated objects is needed. This grid must reflect CMCDOT's occupation grid in the following aspects: origin position, grid direction, cell size and velocities. The bounding box and velocity information provided by CARLA simulator is translated into occupancy grid at each time step to generate the ground truth.

Currently, each lidar is simulated with the appropriate position on the ego vehicle, the same sampling frequency and the same data format as the physical sensor. To match the sensing uncertainty, a Gaussian noise can be added.

In order to be able to efficiently generate a large number of simulated environments, we have perfected a parameter-based approach which streamlines the process through which the dimensions and initial position and velocity of non-ego vehicles are specified.

III. Formal Validation

One of the main assets of CMCDOT is the possibility to estimate the probability of collision for each cell in the grid. The validation focuses on checking the reliability of this risk estimation by performing a rigorous formal verification on execution traces, and then by analyzing the output of the verification using well defined KPI in order to obtain a quantitative evaluation of CMCDOT as grades. Three properties will be checked: *coherence*, *safe prediction* (safety), and *prediction progress* (reactivity).

These properties are defined on execution traces where each line in the trace file is called an event. Each event has a timestamp, the speeds of both vehicles, the probability of collision within 1, 2, and 3 seconds, the positions of both vehicles, and a collision indication. All data are real numbers except the collision indication which is Boolean. We denote $P^i_{col}(e)$ where $i \in \{1,2,3\}$ the probability of collision within i seconds read on event e. We denote $P_{col}(e) = (P^1_{col}(e), P^2_{col}(e), P^3_{col}(e))$ the triple that contains all three probabilities of collision for e.

The verification consists in checking that for every trace T, the set of events e in T that violate each property P is empty:

$$\forall T, T \vDash P \iff \{e \in T \mid e \nvDash P\} = \emptyset$$

Each trace receives a verdict file indicating for each property if the property holds. When it does not, the verdict contains for each property the list of violating events along with evidence of why the event is a violation of P.

The verification is carried out using the XTL [10] model checker of the CADP toolbox [11] dedicated to the formal modelling and analysis of asynchronous concurrent systems. The underlying behavioural models used within CADP are state spaces, i.e., state-transition graphs in which transitions are labeled by events (channel names and data values exchanged) executed by communicating concurrent processes. XTL is both a language and a tool for querying state spaces. It enables the expression of temporal logic operators, observer automata, and to perform more general computations over (sets of) states and transitions, also involving the data values carried by the events. To apply these tools for CMCDOT, we consider the execution traces as particular cases of state spaces consisting of a single finite sequence of transitions, each one being labeled by an event of the trace.

At time t CMCDOT's risk grid indicates for each cell c the probability for c to collide with the ego-vehicle within $t+1$, $t+2$, and $t+3$ seconds. To proceed with our analysis, the probabilities are abstracted into three interpretation classes: *low*, *transitioning* and *high*. For simplicity's sake, these classes are represented by 0, 0.5, and 1 respectively. The thresholds that separate the three classes have been fixed at 0.1 and 0.9.

A. Coherence of the risk probabilities

The three probabilities give overlapping information. In particular, for some event e, $P_{col}(e) = (0, 0.5, 1)$ indicates that no collision should occur during the first second following e but one must occur before the end of the third second following e. This collision may occur during the second second or the third second. However certian configurations like $P_{col}(e) = (1, 0.5, 0)$ are incoherent because they give

Interpretation of $P_{col}(e)$	Meaning in English
(0, 0, 0)	Not before 3
(0, 0, 0.5)	Not before 2
(0, 0, 1)	Between 2 and 3
(0, 0.5, 1)	Between 1 and 3
(0, 1, 1)	Between 1 and 2
(0.5, 1, 1)	Before 2
(1, 1, 1)	Before 1
(0, 0.5, 0.5)	Not before 1
(0.5, 0.5, 1)	Before 3
(0.5, 0.5, 0.5)	Uncertain

TABLE I: Table of coherent combinations of $P_{col}(e)$ (left) and their meaning in English (right) given in terms of the predicted collision. For example, "Between 2 and 3" means that the collision should occur between two and three seconds.

contradictory indications, i.e., there should be no collision within the next three seconds and there must be one within the first second. Altogether there are $3^3 = 27$ combinations for $P_{col}(e)$ where seventeen are incoherent and ten are coherent. In fact, the coherent combinations are only those where the probabilities follow an order relation. Table I lists the ten coherent configurations along with their interpretations. The seven first are *good* configurations while the three last are coherent but *not very useful*. In particular the last one does not give any meaningful information for decision.

The *coherence* property consists in asserting for every event e in a trace T, that the triple $P_{col}(e)$ has one of the ten coherent configurations. This is an invariant assertion that can be checked on every event individually regardless of the information on other events.

B. Safe Prediction

In Table I, the interpretations (left) show that the triple $P_{col}(e)$ is a prediction of *where* a collision should occur or not during the three seconds following e. *Safety* properties specify intuitively that "something bad will never happen" [20]. A *bad* prediction is when $P_{col}(e)$ predicts no collision yet one occurs or when $P_{col}(e)$ does not predict a collision yet one occurs. Safety properties are classic in temporal logic. They will be evaluated with an observer automaton [21], illustrated on Figure 5. For every event e in a trace T, the observer is synchronized with the trace taking as parameter the prediction $P_{col}(e)$. It parses the trace during three seconds following e observing whether a collision occurs or not as predicted. If the automaton decides that all events of T are safe predictions (always ending in the accepting state 4 and never in the rejecting state 5) then T satisfies the *safe prediction* property because "nothing bad happened". This automaton, encoded in XTL and executed on each event of T, computes the set of events that violate this property and produces a verdict for each of them.

C. Prediction Progress

The prediction of collision should progress in two ways as a collision approaches. First, each one of the $P^i_{col}(e)$ ($i \in \{1, 2, 3\}$) should follow the class progression 0, then 0.5, then

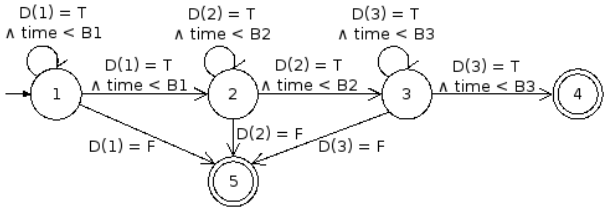

Fig. 5: Safety automaton checking for each event if it is a good or bad prediction. It is a mix between a deciding automaton and a timed automaton. State 1 is the initial state, state 4 is the accepting state, and state 5 is the rejecting state. For a given event e in the trace, the automaton parses the events covering the next three seconds. In state 1 it checks that the collision indication during the first second corresponds to $P^1_{col}(e)$. If so, it moves to state 2, if not it moves to state 5.

1. Second, considering the triple $P_{col}(e)$ altogether, $P^1_{col}(e)$ should reach the value 1 first, then $P^2_{col}(e)$, and then $P^3_{col}(e)$. A collision should only happen after this progression. However this progression can be interrupted by a change of velocity of one of the vehicles. If no collision is ever predicted, $P_{col}(e)$ should remain at $(0, 0, 0)$ for all events.

The *prediction progress* can be stated "once a collision starts being predicted, the prediction will eventually progress until a collision or a change of velocity occur". This is a response property, in which if no collision is predicted, $P_{col}(e)$ should remain at $(0, 0, 0)$ as default behavior, however it adopts a progressing behavior in *reaction* to a collision starting to be predicted. This property expresses that the only valid reason for the prediction not to progress once it started, is a change of velocity. This property is the intersection of a *safety* property in that it waits for the collision to be predicted, and a *liveness* property in that is ensures the order of the progression up to a collision or a change of velocity.

Once again this temporal property is verified using an observer automaton, which will reveal every evolution of the prediction that goes against the expected progression and no change of velocity was observed.

IV. Experiments & Results

A. Scenario description and trace details

Our simulation scenario aims at checking the behavior of cars at a four-way crossroad and validating the Time-to-Collision (TTC) estimated by CMCDOT. For simplification, we bound our simulation with a rule stating that at any given moment in time, a maximum of one simulated non-ego vehicle is present on the crossroad. As shown in Figure 6, the ego-vehicle (white) and non-ego vehicle (red) both start at the same time from a given location in the Carla environment. A PID controller maintains constant velocity for both vehicles and the steering angle is fixed. For simplicity, we only consider the case when both vehicles collide at a 90-degree angle. A scenario ends when the cars collide or the ego-vehicle completes its track, passing the intersection without

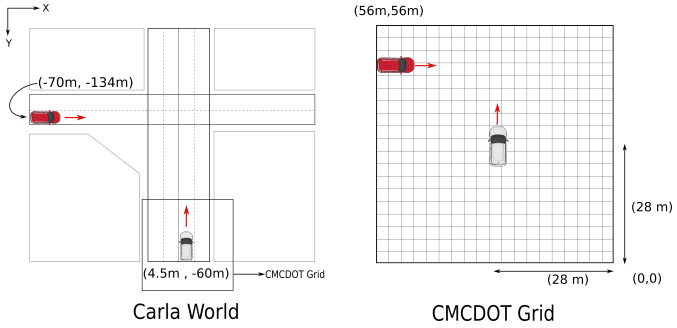

Fig. 6: Intersection Crossing scenario and CMCDOT grid.

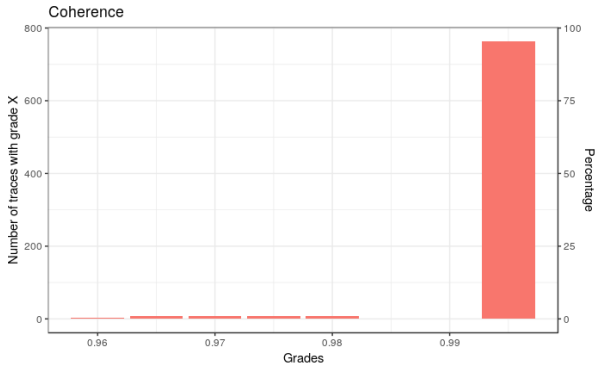

Fig. 7: Coherence results: 95% of the traces have only coherent predictions. 5% of the traces are slightly less coherent. The minimal grade is 0.95.

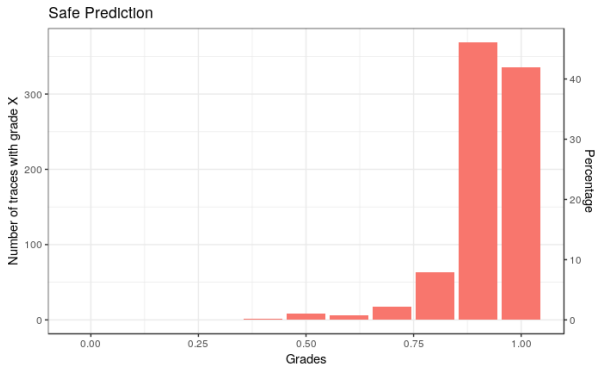

Fig. 8: Safe prediction results: 40% of the traces have safe predictions, 45% generally have safe predictions with the exception of a few events, 15% of the traces have low grades.

any collision. For our study, a total of 800 traces have been considered to generate the results reported here.

To simulate different cases, we rely on the random generation of parameters defining: the non-ego vehicle class (cars or motorcycles), initial positions and initial speeds of the ego and non-ego vehicles. The test cases are then run, and their results (perception output of CMCDOT, as in Fig. 4) are stored with a frequency of 10 Hz alongside the parameter sets. The analysis of these datasets allows us to accurately measure the efficiency of our perception and estimation solution.

The strong advantage of this approach is the ease with which a large number of simulated scenarios can be generated, ran, and analyzed.

B. Validation Results

To assess the validity of CMCDOT's collision prediction, the verification verdicts are analyzed using KPI in order to produce grades as quantitative results. The KPI need metrics, ideal grades, and evaluation functions to give grades to traces w.r.t the metrics and the ideal values. For every property P, each event e in a trace T receives a grade $G_P(e) \in [0, 1]$. The ideal grade is 1 and $G_P(e) = 1$ if e satisfies P. Each property is given metrics to determine the severity of the violations. Upon violation, $G_P(e)$ is reduced by a penalty factor. The grade of a trace is the average of the grades of all its events.

The coherence property verifies the order relation within the $P_{col}(e)$ triple for every event e in a trace. When the order is violated ($P^i_{col}(e) > P^j_{col}(e)$), the penalty for coherence is $P^i_{col}(e) - P^j_{col}(e)$ with $i, j \in \{1, 2, 3\}$ and $i < j$.

Figure 7 shows the grades for the coherence property. 95% of the traces have only coherent predictions. In the remaining traces, none have contradicting predictions. The incoherences that are identified by this property are events similar to a situation with $P^1_{col}(e) = 1$ and $P^2_{col}(e) = 0.99$. Though the order relation is violated, the impact is minimal. Globally CMCDOT is coherent in its predictions.

For the *safe prediction* property, a bad prediction occurring on an event three seconds before a collision has time to be corrected on the following events but a bad prediction occurring during the last second is dangerous for driving. The metrics take this severity scale into account. When an event e violates the safe prediction property, the *safe prediction* grade is $1 - 1/i$, where i designates which of the $P^i_{col}(e)$ violated the property.

Figure 8 shows the corresponding grades. The high grades represent 85% of traces. The grades that are in $[0.9, 1)$ come from executions where the predictions are accurate but one or two events late. The other grades are due to the fact that we are stressing CMCDOT with situations where the vehicles nearly miss or barely touch. The traces with low grades correspond to cases where the collision is predicted but the trace did not indicate a collision, thus giving a bad grade to all the events within three seconds of a collision. This revealed that the bounding boxes that are used in the simulator to determine collisions did not completely fit the vehicle. There were also some very particular scenarios where the vehicles were so close that CMCDOT took longer than usual to predict collisions. In these situations, a human driver would never let vehicles be so close.

Based on these results, we can conclude that CMCDOT satisfies or almost satisfies the *safe prediction* property 85% of the time when confronted to corner cases.

The verdict of the *prediction progress* property lists all the events where the prediction evolved against the intended progression. The penalty for this property is proportional to

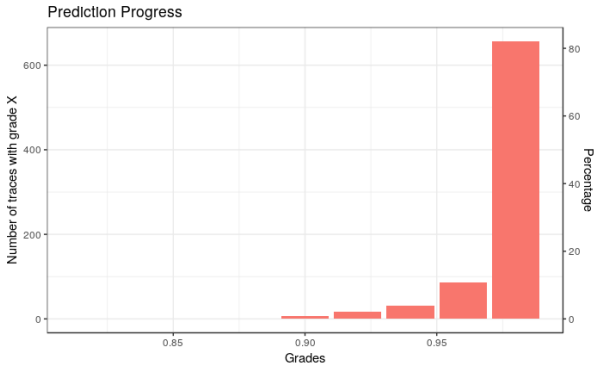

Fig. 9: Prediction progress results: The prediction evolved as expected in 80% of the traces. In some traces CMCDOT hesitated on the progression of the predictions.

how severe the reverse progression is.

Figure 9 shows that for 80% of the traces the predictions progress as intended. In the scenarios there are no changes in the velocities, so the expected behavior is for the predictions to always progress towards a collision. In 80% of the traces, the prediction progresses as expected. The worst grade begins at 0.9, indicating that the traces where the prediction evolves in the reverse order contain punctual fluctuations where CMCDOT hesitates on some predictions. The impact of these hesitations is not significant in the overall evolution of the progression when a collision approaches.

V. Conclusions

We presented a methodology for validating perception systems based on a combination of simulation, formal verification on simulation traces, and statistical analysis of the verification results according to KPIs associated to each property of interest. We instantiated this methodology for validating the collision risk prediction feature of CMCDOT for a set of relevant scenarios involving collisions between vehicles, but also corner cases (near misses or bare touches). Our validation framework, which integrates the CARLA simulator, the CADP verification toolbox, and the RStudio statistical computing environment, enabled us to assess the accurate behaviour of CMCDOT w.r.t. collision risk prediction.

As future work, we plan to take into account the changes of vehicle velocities (variable speeds and changes in direction). Our methodology can be readily leveraged to achieve this, by first pre-cutting the traces into intervals of constant/transient velocities, and then applying the analysis as discussed in Section III. We also plan to consider some more frequent and realistic urban situations, such as stopping behind another vehicle at a stop light or driving for hours to see if CMCDOT still performs well after long executions.

Acknowledgment

This work has also been conducted within the ENABLE-S3 project of the ECSEL joint undertaking under grant agreement NO 692455. This joint undertaking receives support from the European Union's Horizon 2020 research and innovation program and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

References

[1] J. McCarthy, *Towards a Mathematical Science of Computation*. Dordrecht: Springer Netherlands, 1993, pp. 35–56.

[2] H. Garavel and S. Graf, *Formal Methods for Safe and Secure Computer Systems - BSI Study 875*. BSI German Federal Office for Information Security, 2013.

[3] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[4] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, *et al.*, "A perception-driven autonomous urban vehicle," *Journal of Field Robotics*, vol. 25, no. 10, pp. 727–774, 2008.

[5] M. Kamali, L. Dennis, O. McAree, M. Fisher, and S. Veres, "Formal verification of autonomous vehicle platooning," *Science of Computer Programming*, 02 2016.

[6] M. Barbier, A. Renzaglia, J. Quilbeuf, L. Rummelhard, A. Paigwar, C. Laugier, A. Legay, J. Ibañez-Guzmán, and O. Simonin, "Validation of Perception and Decision-Making Systems for Autonomous Driving via Statistical Model Checking," in *IV 2019 - 30th IEEE Intelligent Vehicles Symposium*, 2019, pp. 1–8.

[7] X. Zhao, V. Robu, D. Flynn, F. Dinmohammadi, M. Fisher, and M. Webster, "Probabilistic model checking of robots deployed in extreme environments," *arXiv preprint arXiv:1812.04128*, 2018.

[8] R. Calinescu, C. Ghezzi, K. Johnson, M. Pezzé, Y. Rafiq, and G. Tamburrelli, "Formal verification with confidence intervals to establish quality of service properties of software systems," *IEEE transactions on reliability*, vol. 65, no. 1, pp. 107–125, 2016.

[9] L. Rummelhard, A. Négre, and C. Laugier, "Conditional monte carlo dense occupancy tracker," in *IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 2485–2490.

[10] R. Mateescu and H. Garavel, "XTL: A Meta-Language and Tool for Temporal Logic Model-Checking," in *Proceedings of the International Workshop on Software Tools for Technology Transfer (STTT'98), Aalborg, Denmark*, T. Margaria, Ed. BRICS, July 1998, pp. 33–42.

[11] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, "CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes," *Springer International Journal on Software Tools for Technology Transfer (STTT)*, vol. 15, no. 2, pp. 89–107, Apr. 2013.

[12] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[13] H. Moravec, "Sensor fusion in certainty grids for mobile robots," *AI magazine*, vol. 9, no. 2, p. 61, 1988.

[14] P. Bessière, E. Mazer, J. Ahuactzin-Larios, and K. Mekhnacha, *Bayesian Programming*. CRC Press, 2013.

[15] T. Fortmann, Y. Bar-Shalom, and M. Scheffe, "Multi-target tracking using joint probabilistic data association," in *19th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, vol. 19. IEEE, 1980, pp. 807–812.

[16] Z. Khan, T. Balch, and F. Dellaert, "An mcmc-based particle filter for tracking multiple interacting targets," in *Computer Vision-ECCV*. Springer, 2004, pp. 279–290.

[17] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, Jan 2009.

[18] L. Rummelhard, A. Nègre, M. Perrollaz, and C. Laugier, "Probabilistic grid-based collision risk prediction for driving application," in *International Synposium on Experimental Robotics*, Springer, Ed., 2014.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[20] Z. Manna and A. Pnueli, "A hierarchy of temporal properties." *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, pp. 377–410, 1990.

[21] B. Alpern and F. B. Schneider, "Verifying temporal properties without temporal logic," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 11, pp. 147–167, 12 2001.